Addressing the Big-Earth-Data Variety Challenge with the Hierarchical Triangular Mesh

Michael L. Rilee[1,2], Kwo-Sen Kuo[1,3,4]
Thomas Clune[1], Amidu Oloso[1,5]
[1]NASA GSFC, Greenbelt, MD, USA,
[2]Rilee Systems Technologies, Derwood, MD, USA
[3]Bayesics, LLC, Bowie, MD, USA
[4]University of Maryland, College Park, MD, USA
[5]SSAI, Greenbelt MD, USA
mike@rilee.net
{amidu.o.oloso, kwo-sen.kuo, thomas.l.clune}@nasa.gov

Paul G. Brown
Paradigm4 Inc.
Waltham, MA, USA
pbrown@paradigm4.com

Hongfeng Yu
University of Nebraska, Lincoln, NE, USA
hfyu@unl.edu

***Abstract*—We have implemented an updated Hierarchical Triangular Mesh (HTM) as the basis for a unified data model and an indexing scheme for geoscience data to address the variety challenge of Big Earth Data. We observe that, in the absence of variety, the volume challenge of Big Data is relatively easily addressable with parallel processing. The more important challenge in achieving optimal value with a Big Data solution for Earth Science (ES) data analysis, however, is being able to achieve good scalability with variety. With HTM unifying at least the three popular data models, i.e. Grid, Swath, and Point, used by current ES data products, data preparation time for integrative analysis of diverse datasets can be drastically reduced and better variety scaling can be achieved. In addition, since HTM is also an indexing scheme, when it is used to index all ES datasets, data placement alignment (or co-location) on the shared nothing architecture, which most Big Data systems are based on, is guaranteed and better performance is ensured. Moreover, our updated HTM encoding turns most geospatial set operations into integer interval operations, gaining further performance advantages.**

Keywords-SciDB; HTM; array database; shared nothing architecture; Big Data; variety; Earth Science; DAAC; remote sensing; data analysis; data fusion; load balancing; co-registration; quadtree; indexing; geographic metadata; GIS

I. INTRODUCTION

For a few decades now, the prevailing data practice in Earth Science (ES) has followed a two-step approach: 1) packaging data into files for archival and distribution and 2) cataloging the metadata of the datasets and data files into databases managed by relational database management systems (RDBMSs), thus making data holdings discoverable and searchable. The establishment of distributed active archive centers (DAACs) as data warehouses and the standardization of data file format through the HDF/netCDF [1] Application Programming Interface (API) by NASA Earth Observing System Data Information System (EOSDIS) since the 1990s exemplify the apex of this approach. These DAACs have since enjoyed much popularity. Consequently, this approach has become the de facto standard paradigm and has been adopted by other organizations.

Although this two-step approach has elevated the convenience of finding, obtaining, and using data to unprecedented heights, it has obviously reached its limit when faced with today's demands. Its limitation is a consequence of the fact that users cannot access and manipulate the data directly but have to work with and through their containers (i.e., files) requiring specialized expertise and resources beyond scientific analysis. This approach has historically led inevitably to data download (mostly through low-bandwidth Internet connections) and integration for analysis by users. With few exceptions, it is still largely practiced today with the same consequences.

Since, as a norm, users cannot access and analyze the data "in place", they must follow the FTP links that result from the metadata search and proceed to download the data files before analysis may commence. In preparation for the download and subsequent analysis, however, users (or their institutions) must first procure necessary compute and storage resources and then address the associated cost of management and maintenance. Moreover, researchers now must also engage in data management activities, such as organizing and backing up downloaded data. Then, they have to familiarize themselves with the organization and meanings of the data elements in the files. These activities, irrelevant to research pursuits, unnecessarily encumber researchers and distract them from their investigations, hampering productivity. The most disheartening aspect of this practice is perhaps the collective waste it causes because almost every data analysis research endeavor needs to duplicate the process and the resources it requires.

Once the downloaded data become "local" to individual research users or institutions, their processing is subjected to the preferences of these individuals (e.g., data management policies or the choices of programming languages for data analysis). The profusion of these preferences, in turn, erects nearly insurmountable barriers against effective cross-institution and/or cross-disciplinary collaborations. In addition, since most geoscience researchers are not professional software engineers and rarely follow good software engineering practices, such as performing unit (and other) tests and subjecting source code to version control, software quality becomes suspect and research reproducibility becomes illusive.

A. *Volume vs. Variety*

Since supercomputing centers have achieved remarkable success in maximizing value for computer simulations, it is natural that the first attempts to address the data analysis challenges described above are to leverage the existing supercomputing infrastructure and architectures. This category of solutions is characterized mostly by moving the data files to the dedicated, but expensive, file systems of the supercomputing facilities, where computing resources are readily available close to the data.

Although good scaling in *volume* can be achieved with this approach, it is a very different story with *variety*. The diverse variety of ES data arises mainly from the multiplicity of observations. "In situ" and "remotely sensed" are the two main categories of observational data. Remote sensing data can again be categorized into ground-based (e.g., weather radars), airborne, and space-based depending on the instruments' locations/positions relative to Earth's surface. The instruments used in remote sensing can also be divided into passive sensors (e.g., radiometer and imager) and active sensors (e.g., radar and lidar) according to their operation modes. These instruments—in accordance with their positions, purposes, requirements, and physical or practical constraints—often utilize different sensing geometries and observe with different resolutions, both spatially and temporally. Further processing of these data generates even greater varieties.

B. *Variety's Toll on Iterative Analysis*

When an integrative analysis requiring diverse datasets needs to be performed, the data preparation effort is not substantially reduced even with the aforementioned supercomputing solution with its nearby compute resource, because each file in each variety has to be processed and homogenized first before the integrative analysis can continue. Unless the "fused" data of the integrative analysis is saved, anyone who wishes to perform the same analysis will have to repeat the entire, same data preparation process. If a similar, but not exactly the same, integrative analysis needs to be performed, e.g. using different spatiotemporal subsets or with slightly different data products, data preparation almost always has to be redone; results and intermediate products from previous efforts can scarcely be reused. Since scientific analysis is characterized by continuous reanalysis, the costs of data preparation exert an expensive toll at every iteration.

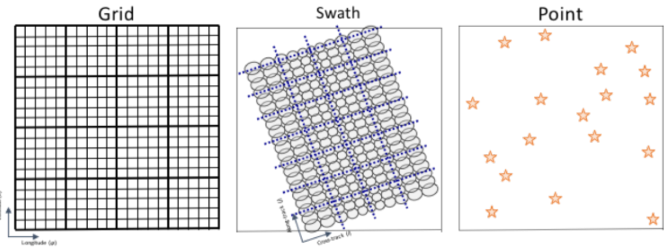

Figure 1. Three data models.

C. *In-place analysis via Partition Placement Co-alignment*

Big Data technologies help enable in-place analysis, including the creation and sharing of data products, yet ES data places special demands on computing architectures. In geoscience data analysis, *spatiotemporal coincidence* is a fundamental requirement. For example, when analyzing cloud formation, we need other spatiotemporally coincident information, such as temperature, pressure, humidity, airflow, etc. Yet when distributed arrays of ES data with misaligned partitions need to be analyzed together, the analysis has to perform computationally expensive repartitionings on the fly to move and align the data first [2], degrading overall performance. When the placements of ES data partitions are co-located on a shared-nothing architecture (SNA), better overall performance can be achieved. Currently this movement occurs when an end-user integrates data on their own, incurring problems mentioned previously. Partition Placement Co-alignment (PPC) is required to realize in-place analysis on distributed architectures, especially when data transfer and integration is costly, further motivating SNA.

D. *A Unifying Approach*

We are developing an in-place analysis system for ES data analysis and sharing. We are building on SciDB, an analysis environment that scales to the massive, distributed, parallel systems that are required to integrate and analyze diverse, voluminous ES datasets [3,4]. SciDB is based on the SNA which is exceptionally suited for pleasingly (aka embarrassingly) parallel computations (i.e., distributed parallel computations mostly exempt from inter-process communication). Inter-process communication is synonymous here to data movement, albeit movement within the SNA cluster. The most ideal PPC strategy is thus one that renders the greatest majority of analyses pleasingly parallel. As mentioned above, SciDB also supports tightly-coupled scientific calculations that are not pleasingly parallel.

Because they are fundamental to ES integration and analysis, we are adding geometric functions to the SciDB array database and scientific calculation platform [5]. We build on the work of Szalay et al. who developed the Hierarchical Triangular Mesh (HTM) and applied it to index the Sloan Digital Sky Survey (SDSS) [6,7]. HTM's efficient indexing and fast integer representation provides a common geographical reference for comparing and combining different data sets. HTM's quad-tree-based indexing scheme

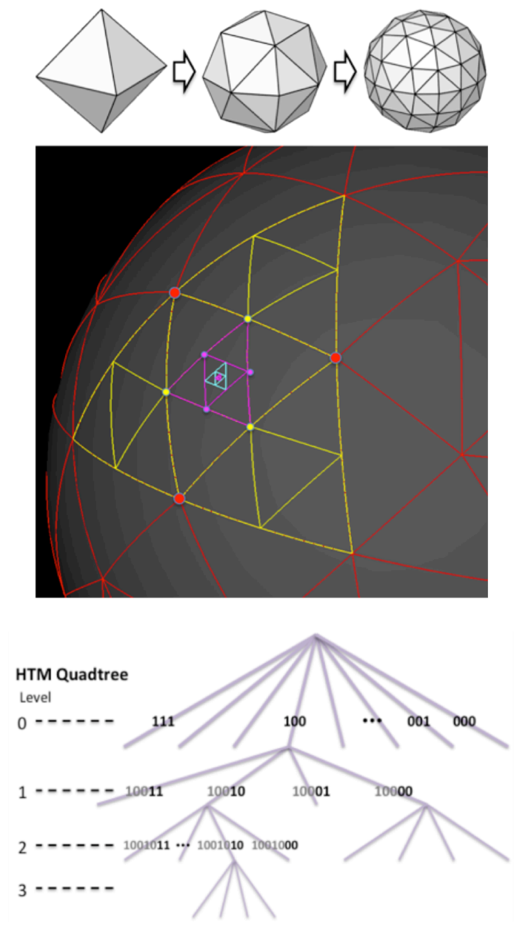

Figure 2. HTM: Projecting the octagon onto and recursively partitioning the sphere into a quad-tree, whose branches are labeled in binary.

recursively subdivides triangles into four children by bisecting the parent's edges, starting with a parent spherical octahedron. This indexing scheme leads to an efficient mapping of the sphere to integer intervals, which we then use for indexing, search, intersection, conditional subsetting, co-registration, regridding, among other tasks, including critically, PPC for load balancing and minimizing data transfers.

In this work we describe a new "Left Justified" bit format that is more appropriate to indexing and integrating distributed ES data. We discuss how our new hierarchical labeling is used for the efficient distribution of data across computational nodes. Temporal indexing and other forms of variety are important but relatively simpler due to its one-dimensional nature and beyond the scope of this paper.

II. Dealing with Diversity: Point, Swath, & Grid

The spatial aspect of geoscience data variety can generally be represented by three data models with array as the data structure: Grid, Swath, and Point (Fig. 1). Grid is a mesh with fixed latitude and longitude spacing and thus a simple linear relation exists between array indices and latitude-longitude geolocation coordinates. Swath retains the spaceborne instrument's observation geometry (e.g., cross- × along-track), where no simple relation exists between array indices and geolocations. Rather, geolocation is specified individually for each Swath array element, i.e. Instantaneous Filed of View (IFOV). The Point model is used mostly for in situ observations made at irregularly distributed locations, which are encased in a vector (1D array). Similar to Swath, geolocation for each point is also specified individually. The dissimilarities among these data models give rise to difficulties in integrative analysis. For example, simply determining the (approximate) common area covered by two Swath arrays can become algorithmically involved, especially if the swaths are from satellites with different orbit characteristics.

These data models, however, have one commonality: data values associated with geolocations, which can thus serve as a basis for a unified data model. Indeed, at the heart of our unified data model is an indexing scheme that essentially assigns an "address" (index) to every surface element (up to a desired resolution) of Earth (i.e., geolocation), e.g. Fig. 2. When data are stored in arrays indexed by this address, it allows quick retrieval of data values associated with any given geolocation, regardless of their original data models. A good index like HTM speeds the comparison and integration of data with different geometries (Fig. 3).

III. Hierarchical Spherical Triangular Mesh

Details about HTM indexing have been described elsewhere [5-13]. The HTM is based on the recursive quadfurcation of a root spherical octahedron, which is divided into South and North halves labeled with a 0 and 1, respectively, Fig. 2. The 4 triangles of each half are labeled 0-3 starting from the triangle nearest the x-axis proceeding counterclockwise around the sphere as viewed from outside and above the poles. At each following level of recursion, the 4 child triangles constructed by adding a triangle connecting the midpoints of an existing triangle's edges are labeled 0-3 according to order in which the parent's vertices are stored.

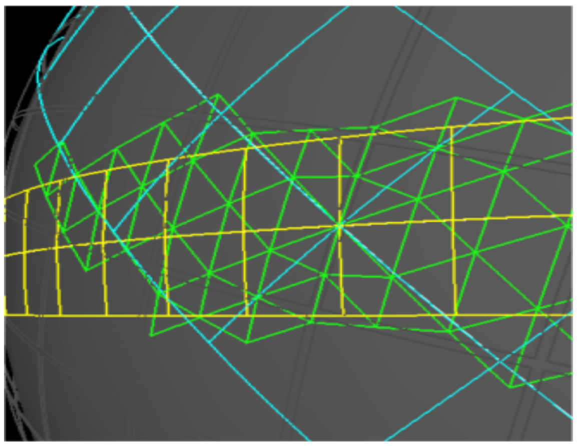

Figure 3. Finding spatial intersections using the HTM.

The symbolic form of the HTM appends the label for each level, with each digit describing how to traverse the quad-tree structure from the root octahedron to the leaf triangle. For example, the 2nd child (a grandchild) of the 1st child of the first (0th) triangle counterclockwise from the x-axis in the northern hemisphere would be denoted N012, a level 2 triangle; examples of the symbolic labeling of triangles is illustrated in Fig. 4.

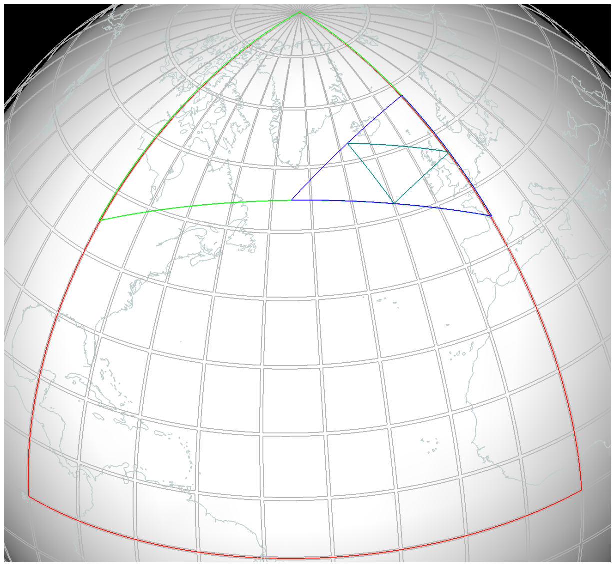

Figure 4. The spherical triangles N0 (red), N01 (green), N012 (purple), N0123 (cyan).

A. *Mapping HTM indices to integers*

As pointed out by Gray, a triangle with an index contains all of the points inside it, completely covering that triangle [8]. One can map HTM symbolic names, and hence the indexed triangular regions, to integers in a number of ways. As can be seen with the symbolic form described above, triangles that share the same "prefix" are children of the triangle denoted by that prefix. For example (Fig. 4), with their shared prefix in boldface, the triangles

N0123123 and
N0123333

are both contained in the triangle

N0123.

Note that the triangles contained within a parent are readily labeled using the parent's representation as a prefix. For example all of the level 6 children of the level 3 triangle N0123 are in the range

N0123000 – N0123333

without any gaps in the coverage. Regions on the sphere may be approximated or covered by sets of these HTM indexed triangles. When encoded as integers, contiguous sequences of HTM indices corresponding to a geographical region can be replaced by integer intervals, such as in the previous example. Searching and geometric calculations such as calculating intersections between datasets are made more efficient by substituting integer operations for the floating-point spherical trigonometry or 3D vector geometry calculations. This dramatic reduction in computational effort and storage enables the provision of detailed geometric information as HTM geometric metadata, accelerating the process of integrating different datasets. This is a crucial reason why we are adding HTM support to SciDB.

B. *Right Justified HTM integer index*

The SDSS/HTM implementation used a straightforward mapping from the symbolic representation mentioned above to integers. In their representation, zero, 0x0 in hexadecimal, is the invalid HTM index. For valid indices, a top or depth bit was set and succeeding bits were set according to the symbolic representation with S and N being represented by 0 and 1 respectively. With the prefix 0b indicating a binary representation, we have the following example mappings of the HTM symbols to integers. We call this a Right Justified Mapping (RJM) or format.

```
S0123  -> 0b1000011011   = 0x21b = 539
N0123  -> 0b1100011011   = 0x31b = 795
S01230 -> 0b100001101100 = 0x86c = 2156
```

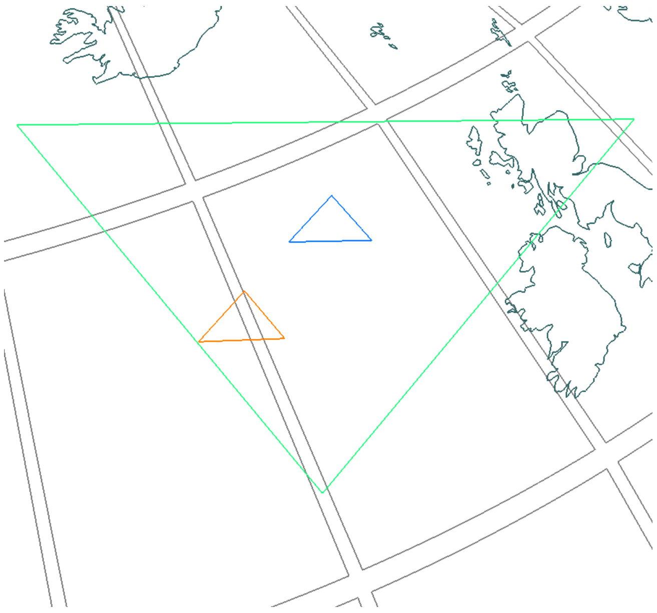

Figure 5. Triangles sharing a prefix are contained in the parent with that prefix. Here N0123 (green) has children N0123123 (orange), and N0123333 (blue).

Unfortunately, RJM maps points in geometric proximity (e.g. in the offspring triangles of the same parent) to multiple, separated locations on the number line. Note that geometrically S0123 (corresponding to the digital value 539 RJM) contains S01230 (2156 RJM), but that when mapped to integers N0123 (795 RJM) it lies in between, even though S0123 and S01230 share the same prefix to the 3rd level (Figs. 5). Thus the correspondence between HTM regions and RJM integer intervals holds only within the same HTM index levels. Implementing geometric set operations, e.g. intersection, under RJM is complicated by this one-to-many mapping of the geometric points (in triangles) along the number line. HTM indices in RJM at a given level form a contiguous sequence of integers, but our diverse datasets feature a wide range of spatial resolutions.

For example, a data set with 5-km footprints might be well suited with level 11 HTM triangles, while a 150-km footprint might get by with level 6. In most cases we are more interested in groups of measurements, e.g. a swath of data as opposed to individual measurements at the IFOV level, for co-location and regridding over a geographic region. Therefore we desire to use triangles from multiple levels of HTM to more succinctly approximate and index these regions. As with other adaptive gridding schemes, large triangles in the interior of such regions are supplemented with smaller triangles in the vicinity of the region's boundary, leading to compact descriptions, all mapped to integer intervals (Fig. 6).

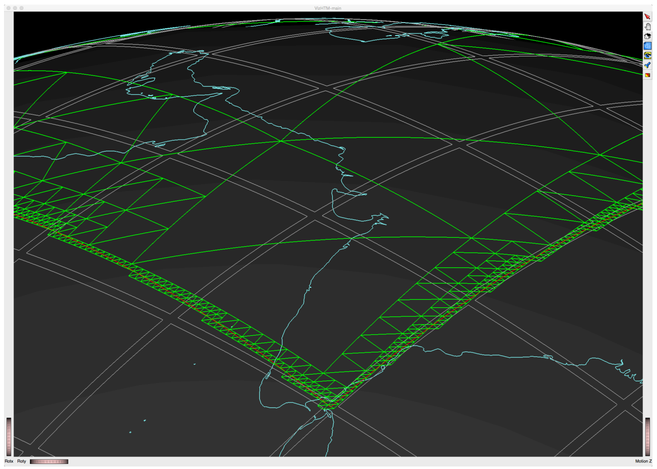
Figure 6. Approximating a region with triangles at multiple resolution levels.

C. Left Justified Integer Intervals

So far we have discussed integer intervals as an optimization scheme for sets of HTM indices. We have noted that some geometric set operations have analogous integer operations, though complicated by the RJM encoding. If indexing were our only goal, we could live with RJM. But because we require a better correspondence between HTM integers and geometry to support data locality for distributed scientific analyses as well as for efficient geometric metadata set operations, we thus must revisit the HTM encoding.

A more convenient mapping to the integers indices that allows other tools to take advantage of the HTM's geometry using integer operations without necessarily falling back on HTM's floating-point geometry, is to use a Left Justified Mapping (LJM) bit format, instead of the RJM. In this representation the above example becomes

```
S0123  -> 0b100001101100  = 0x86c = 2156
N0123  -> 0b110001101100  = 0xc6c = 3180
S01230 -> 0b100001101100  = 0x86c = 2156
```

using 12-bit words for clarity.

Now we see that the common HTM prefix is rendered into the integers in the same way for different HTM levels, so now geometric containment is respected by the mapping. Unfortunately, as demonstrated in the above example, we now have an aliasing problem, e.g. with S0123 and S01230, in that the LJM has difficulty distinguishing between levels. In other words, just how many bits from the left are significant? For the RJM, the top or depth bit tells you both the level of the HTM index and where the significant bits start. To complete the LJM approach we need to do three more things.

First, we need to keep track of the level. In a programming environment like C++ we can merely keep track of the level in another field in a struct or an object. But since we are mapping to integer intervals so other tools may readily take advantage of the implicit geometric encoding, we must improve the mapping. For this discussion we confine ourselves to 64 bit integers, moreover to signed 64 bit integers for technical reasons which may be relaxed if needed. To track the quadfurcation level we devote the rightmost (least significant) 6 bits, using the rightmost 5 for the actual level number and the remaining bit (the 6th) in reserve. We also reserve the leftmost bit for internal use and set the remaining bits as in the left justified example above--except we drop the top/depth bit as being unnecessary.

Table 1. Left Justified Mapping

Bit position	Use
most significant 63	Reserved // Top Bit
62	North-South Bit
60..61	Octahedral triangle index Resolution level 0 ~10 km
6..59	Quadtree triangle index Resolution levels 1-27 ~5 km to ~7 cm
5	Reserved // Terminator Bit
0..4 least significant	Resolution level // Terminator

Thus the triangles from our previous example become:

```
S0123  -> 0x06c0000000000003
S01230 -> 0x06c0000000000004
N0123  -> 0x46c0000000000003
```

This almost gets us where we need to go. We can tell the difference between S0123 and S01230 and integer comparisons can tell us that the latter is contained in the former, but cannot distinguish the reverse. Therefore the inclusion of levels in the integers respects the underlying geometric meaning of the HTM triangles.

Note that the difference between these integer representations of S0123 and S01230 is merely the difference in their resolution levels. Since the maximum resolution level is 27 for our 64 bit integer encoding (see Table 1 above), for any two HTM integers that differ by 27 or less, the triangle associated with the lesser will contain the triangle associated with the greater. In fact, traversing through the levels for these otherwise identical integers corresponds to traversing a quad-tree by always choosing the 0th triangular partition at a particular resolution/quadfurcation level. Next we deal with the integers that are greater than this special sequence but less than the next indexed triangle at the same quadfurcation level.

The second step is to understand how triangles of different levels are interleaved in LJM and the role that integer intervals play. There are natural upper and lower bounds to the set of all labeled child triangles within a given triangle. For the lower bound one merely takes the HTM index of that given triangle as a prefix and then appends zeros down to the maximum allowed resolution of the representation (excluding the 6 bits reserved for the quadfurcation level), i.e. one needn't change the current representation. Consider the following consecutive level 3 triangles indexed and mapped as follows.

```
lower bound S0123 0x06c0000000000003
upper bound S0130 0x0700000000000003(faulty)
```

Ignoring the 6 bits reserved for the quadfurcation level, the numbers between these two limits correspond to all of the triangles in S0123, i.e. traversals of the HTM quadtree from that triangle, representable in our 64 bit LJM. Using S0130 as an upper bound for triangles in S0123 is problematic, because of the aliasing problem. For performance considerations it is very desirable that integer order operations, e.g. "<" or "<=", could be made to stand in for geometric intersection. However, special care must be taken for the faulty upper boundary LJM representation above, because one has valid HTM integers less than the upper bound S0130 above, but still not in S0123, e.g. S013 at level 2. This complicates the code surrounding inclusion at the upper bound, and we can do better.

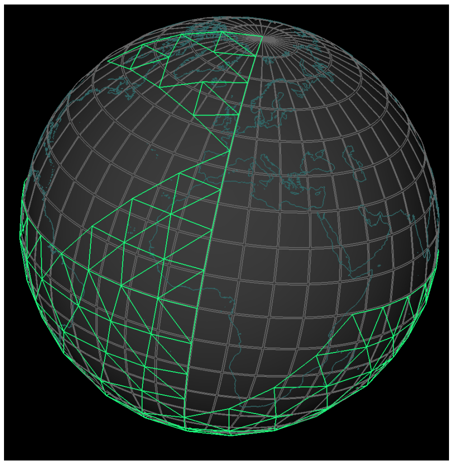
Figure 7. The level 3 interval S0123..N0123 with multiple triangles. Otherwise filled, the interval has a geometric hole, the larger level 2 triangle N013 near the pole.

The third thing to do is thus to address the upper bound: if we used the encoding scheme discussed above, drawing a correspondence between integer interval operations and geometric set operations becomes troublesome. Some of these problems are lessened if we use the last, smallest indexed triangle as the upper bound, which are as follows.

```
upper bound S0123  -> 0x06c3ffffffffffc3
upper bound S01230 -> 0x06c3ffffffffffc4
```

Where we have essentially selected the 3rd triangle at each quadfurcation (adding ones at each bit position down to the rightmost 6 bits). However, this representation again confuses the logic associated with determining inclusion at the upper bound. One could mask off the level bits or make use of the unused 6th bit, but it is easier to introduce a terminator for the interval by simply setting all bits to the right of the significant HTM location bits to one. For a 6-bit field, this corresponds to the number 63, 0x3F. Level information is already contained in the lower bound for an interval and is thus redundant in the upper bound. Therefore, with intervals, the above examples become:

```
S0123  0x06c0000000000003-0x06c3ffffffffffff
S01230 0x06c0000000000004-0x06c3ffffffffffff
N0123  0x46c0000000000003-0x46c3ffffffffffff
```

With this mapping the integer interval mappings of all of the triangles in S01230 fall within the interval `0x06c0000000000004-0x06c3ffffffffffff`, and intersections can be performed with integer operations.

If we were only dealing with individual triangles, we needn't explicitly save the terminator along with the lower bound, but the terminator is important when the interval is for more than one triangle at a given level.

An example of an interval including multiple triangles at the same level is

```
S0123-N0123
        0x06c0000000000003-0x46c3ffffffffffff
```

corresponding to a complex region on the sphere (Fig. 7). Note that all of these triangles in the interval are at the same quadfurcation level. When triangles at different levels are combined in one set of intervals, one must fix how one handles areas of overlap. For our searching and co-registration needs, subsuming higher resolution triangles into overlapping lower resolution intervals is appropriate, though it adds the complexity of having to edit the higher resolution intervals when intersections occur when adding intervals. For example, in Figs. 4 and 5 the smaller triangles would be subsumed in their parents.

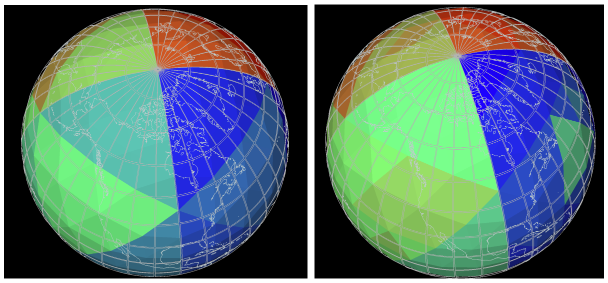
Figure 8. Simple automatic chunking based on HTM integer intervals for different spatial distributions of data/work. Left: homogeneous. Right: non-uniform. Different colors correspond to different partitions and their placement.

The mapping from HTM to integer intervals described above tends to map nearby portions of the sphere to nearby portions of the number line. Since the indexing is based on a quadratic tree, it is possible for neighboring leaf nodes (triangles) to have no parent nodes in common, save for the root. In this case, neighboring triangles (in a geometric sense) could correspond to integers that are rather far apart (in a numeric sense). This occurs, for example, at the boundaries of the spherical triangles making up the root octahedron. So whereas intersections are respected by the mapping, finding geometric nearest-neighbors via integer intervals is not trivial and is much easier using the quad-tree traversal functions of HTM.

IV. Applying the HTM on SciDB

The left justified interval mapping plays a critical role in adding geospatial capabilities to SciDB. SciDB is based on a distributed/parallel shared-nothing paradigm for its array database functions as well as its MPI-based analysis functions. Shared-nothing means that data is not redundantly shared across the SciDB compute nodes and is not communicated from node to node to the greatest extent possible. The LJM provides a natural way to partition the sphere and distribute data across computational nodes (Fig. 8). Once one constructs HTM index interval metadata for the datasets of interest, we can then partition those intervals into a new set that balances the amount of data across the compute nodes. That is, each compute node is associated with its own set of HTM index intervals, and hence geometric region on the sphere. Consequently geometric operations such as intersections on the data are balanced across those nodes, communications are minimized because geometrically local operations are kept local on the compute nodes. This linkage of geometric and computational partitioning is essential for realizing the goals of efficiently using diverse, large scale Earth Science data in SciDB in place.

We add our updated HTM to SciDB using its User Defined Type and Function (UDT and UDF) facilities and have constructed a preliminary HTM UDT verifying the basic functionality of the type and related functions. Our extension can construct the new left and the old right justified mappings as well as the symbolic names of HTM triangles for use as indexes in SciDB's arrays.

The SDSS/HTM we have extended has an efficient and fast skip list-based implementation of sets (called ranges in the code) of HTM integer intervals for handling complex geometric regions [14]. We are continually adding geometric set operations so that we may eventually handle a broad range of geographic regions. As these are integrated with SciDB we will demonstrate and measure the performance of HTM-enabled geometric functions on full scale Earth Science data for our NASA-funded DERECHOS project, which uses multiple sources of data to automatically identify spatio-temporally extended weather events, starting with blizzards.

At this point completing the integration of HTM with SciDB is straightforward. More interesting are the capabilities that will be demonstrated when we start to tag Earth Science data with HTM ranges, sets (arrays in SciDB) of our left justified integer intervals. This will allow us to geometrically compare, co-register, and select diverse kinds of data via metadata operations. The HTM gridding, as a standardized intermediary, provides extra options for quantitatively combining data.

Rather than depending on the assumptions made by stovepiped producers of high-level gridded data products, a researcher can work closer to the measurements and uncertainties, with lower level data, relying on the HTM indexing in the SciDB array database to help automate handling the data's diverse geometries. We are also working on a regridding capability in which the system uses the HTM to construct grids adapted to the intrinsic geometry of the data itself, whether gridded, unstructured, or swath, and then provides the means to transform the data from one grid to another. Researchers will be able to choose, according the needs of their analysis, whether to regrid data to the grid associated with a particular sensor or to a different grid not associated with any particular data source. As an aside, the triangles used in the HTM, being quasi-equiareal, do not suffer the extreme singularities and distortions of the popular lat-lon grids, yet we plan to take advantage of the Earth Science community's support of a variety of grid geometries (e.g. geodesic and CubeSphere grids) so that our HTM-based functions help automate the interoperability of existing grids and the data driven grids HTM enables.

SciDB provides a compute and storage paradigm better suited to the large-scale distributed data intensive Earth Science than the traditional 2-step of download and integrate, even when supported by supercomputing centers. Its shared nothing architecture limits costly data transfers and opens up "in place" analysis and data sharing bypassing expensive data preparation and integration. The new HTM with its left justified bit format provides an efficient uniform geographic platform for integrating diverse datasets, enables efficient implementation of the shared nothing architecture via data partition placement co-alignment, and provides a compact representation for geographic metadata. Thus SciDB and the new HTM point the way towards a data analysis environment that scales to the current Big Earth Science Data analysis variety challenge.

V. ACKNOWLEDGMENTS

Support for this work is primarily provided by the NASA Earth Science Technology Office's Advanced Information System Technology program and partially by the National Science Foundation's EarthCube program.

REFERENCES

[1] The Network Common Data Form (NetCDF): http://www.unidata.ucar.edu/software/netcdf/; The HDF Group, Hierarchical Data Format: http://www.unidata.ucar.edu/software/netcdf/

[2] Clune, T., K.-S. Kuo, K. Doan, and A. Oloso. 2015. "SciDB versus Spark: A preliminary comparison based on an Earth science use case," in *AGU Fall Meeting*, San Francisco, California, 2015.

[3] SciDB by Paradigm4: www.paradigm4.com; J. Rogers et al. Overview of SciDB: Large scale array storage, processing and analysis. In SIGMOD, 2010.

[4] Brown, P. 2015. SciDB and Geoinformatics Analysis. Geophysical Research Abstracts, Vol. 17, EGU2015-15102, 2015. EGU General Assembly 2015

[5] The Hierarchical Spherical Triangular Mesh (HSTM) website: https://github.com/michaelleerilee/hstm

[6] Szalay, A.S., Gray, J., Fekete, G., Kunszt, P.Z., Kukol, P., and Thakar, A. 2005. Indexing the Sphere with the Hierarchical Triangular Mesh. Microsoft Research Technical Report, MSR-TR-2005-123.

[7] The Sloan Digital Sky Survey and HTM websites: http://skyserver.sdss.org/ and http://www.skyserver.org/htm/

[8] Gray, J., Szalay, A.S., Fekete, G., O'Mullane, W., Santisteban, M.A.N., Thakar, A., Heber, G., Rots, A.H. 2004. There Goes the Neighborhood: Relational Algebra for Spatial Data Search. Microsoft Research Technical Report, MSR-TR-2004-32.

[9] Fekete, G., Kuo, K.-S. 2015. Indexing Earth with Trixels. Poster presentation at the 8th XLDB Conference, May 19-20, 2015 Stanford University, CA, USA.

[10] Kunszt, P.Z., Szalay, A.S., Thakar, A.R. 2001. The Hierarchical Triangular Mesh. In Mining the Sky: Proceedings of the MPA/ESO/MPE Workshop held at Garching, Berlin/Heidelberg, 2001, Ch. 83, p631.

[11] Barrett, P. 1994. Application of the Linear Quadtree to Astronomical Databases, Poster at ADASS 1994.

[12] Fekete, G. 1990. Rendering and managing spherical data with sphere quadtrees. Proc. of Visualization '90. IEEE Computer Society, Los Alamitos, CA. pp. 176-186.

[13] Crichton, D.J., Mattmann, C.A., Cinquini, L., Braverman, A., Waliser, D., Gunson, M., Hart, A.F., Goodale, C.E., Lean, P., and Kim, J. 2012. Sharing Satellite Observations with the Climate-Modeling Community: Software and Architecture. IEEE Software, September/October 2012, pp. 73-81.

[14] Pugh, William. 1990. Skip lists: a probabilistic alternative to balanced trees in Communications of the ACM, June 1990, 33(6) 668-676.